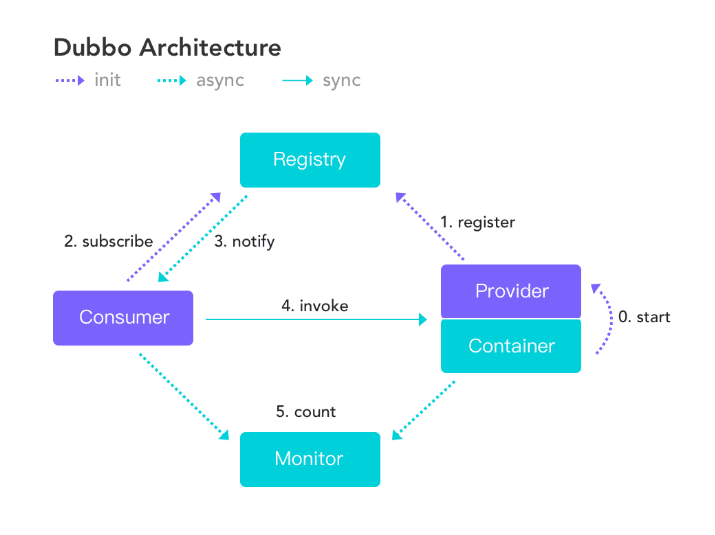
# 一、搭建注册中心



## 1.zk的注册中心

zookeeper是java写的，因此运行zk的服务的机器需要安装jdk的。

上传zookeeper的压缩包到服务器上

## 2.解压缩并编写配置文件

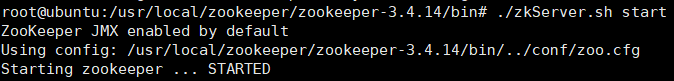
在conf/zoo\_sample.cfg==重命名==> zoo.cfg

## 3.进入到bin，启动zk

操作zk的相关命令

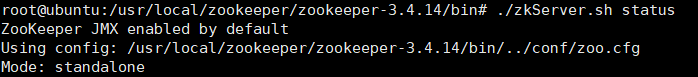
### 1） ./zkServer.sh start

启动zk



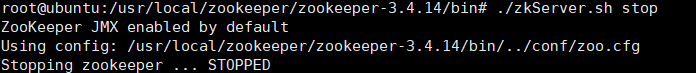
### 2） ./zkServer.sh status

查看zk的状态



### 3） ./zkServer.sh stop

停止zk



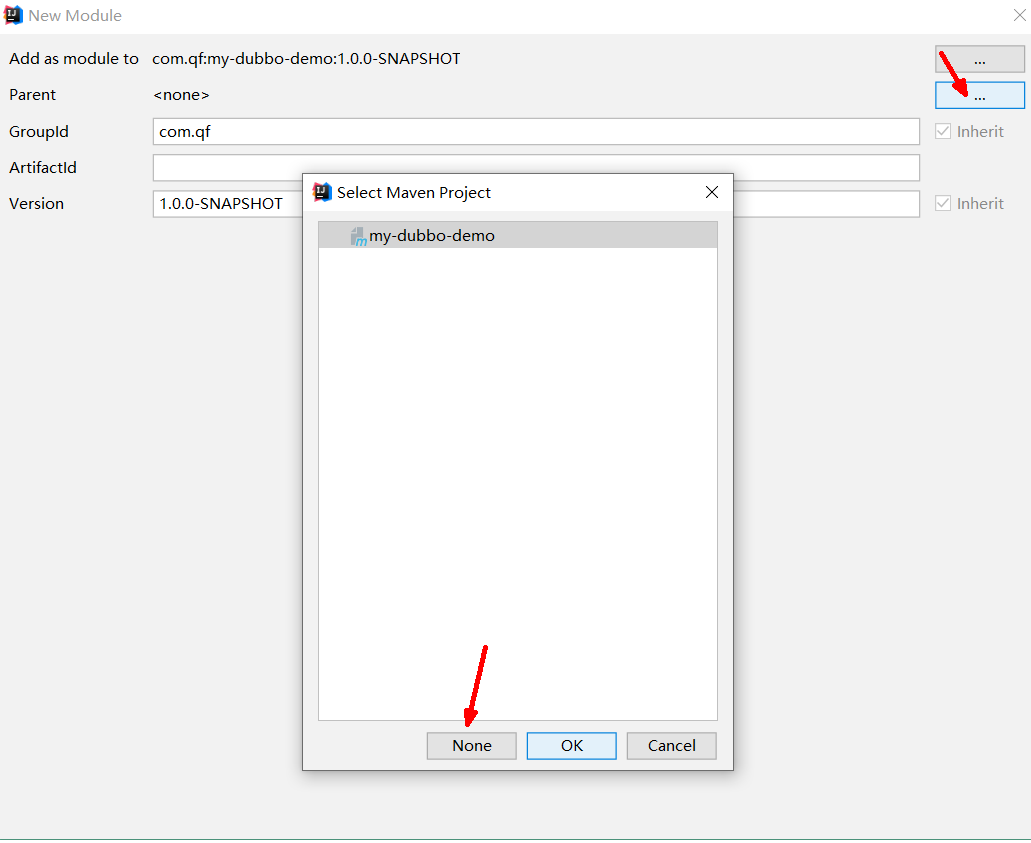
# 二、创建第一个Dubbo的demo

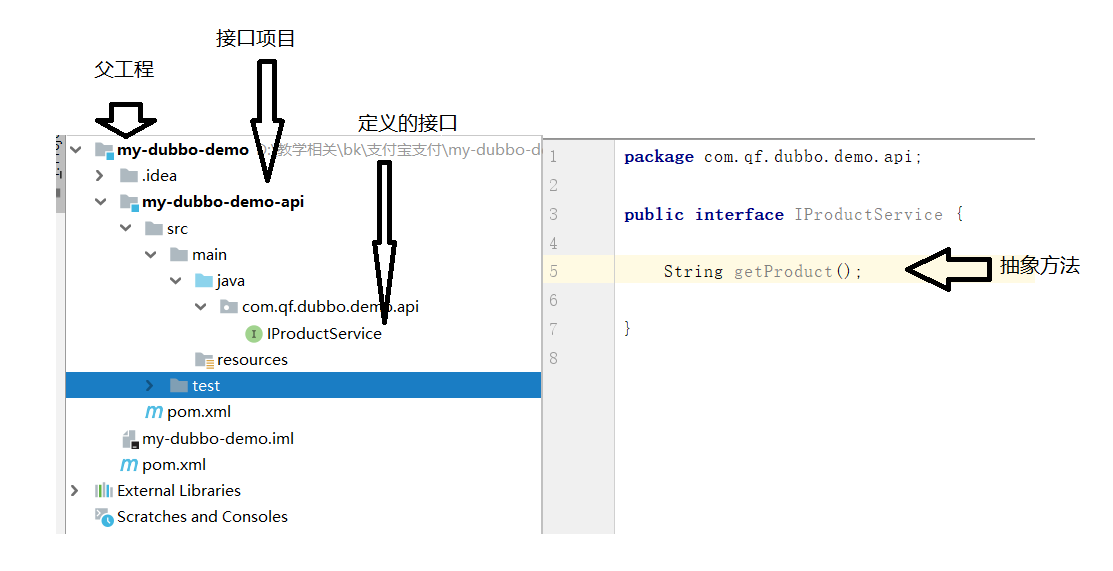
## 1.创建一个父工程



## 2.在父工程中创建接口项目

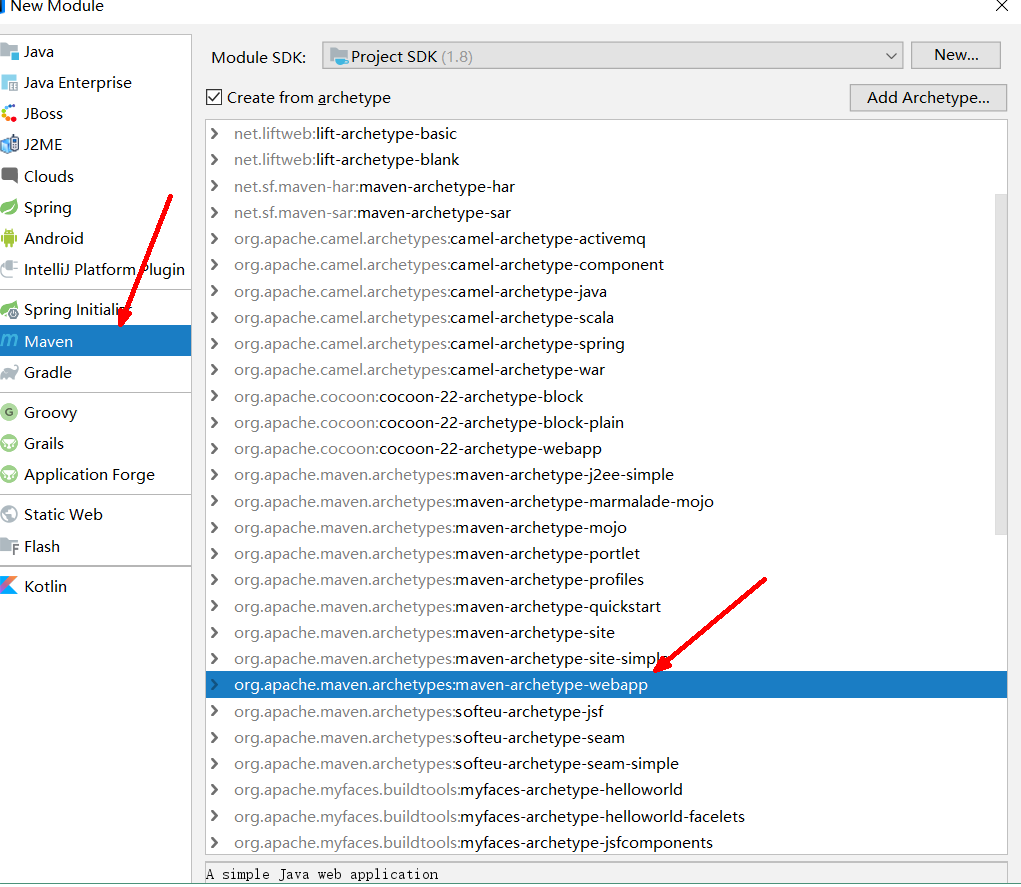
让接口项目没有父项目的依赖



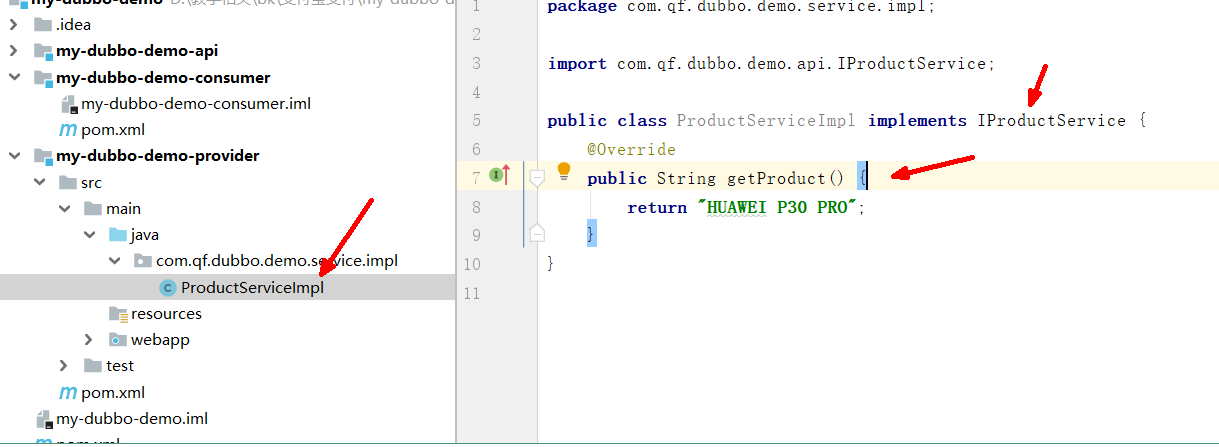


## 3.创建服务提供者项目

1）它是一个maven的webapp骨架项目



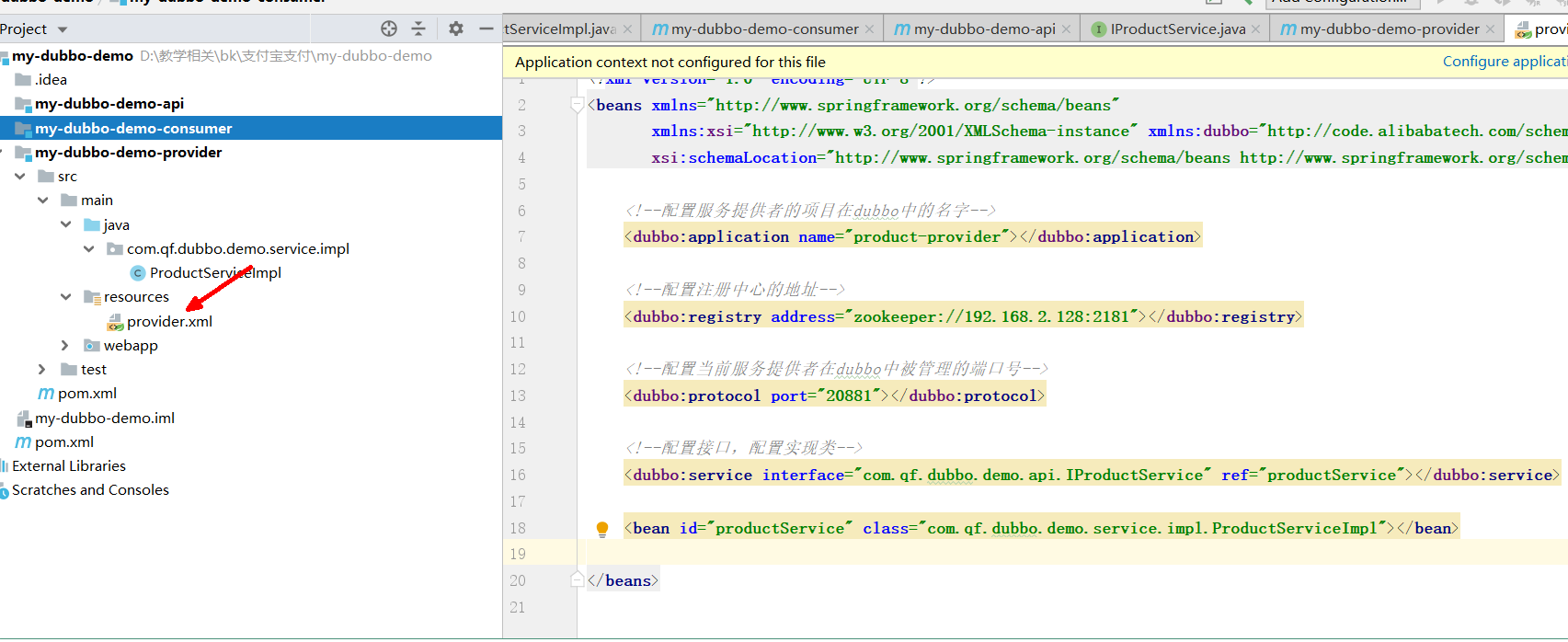
2）编写接口的实现类，重写抽象方法



3）引入dubbo和zk的依赖

<**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>dubbo</**artifactId**>  
 <**version**>2.5.8</**version**>  
</**dependency**> *<!-- zookeeper客户端依赖 -->*<**dependency**>  
 <**groupId**>com.github.sgroschupf</**groupId**>  
 <**artifactId**>zkclient</**artifactId**>  
 <**version**>0.1</**version**>  
</**dependency**>

4）编写配置文件



5）将dubbo中服务注册到注册中心上

通过测试方法，将服务注册到注册中心

|  |
| --- |
| @Test **public void** test1() **throws** IOException {   ClassPathXmlApplicationContext context =  **new** ClassPathXmlApplicationContext(**new** String[]{**"classpath:provider.xml"**});  context.start();  System.***out***.println(**"服务已注册到注册中心"**);  System.***out***.println(**"按任意键停止"**);  System.***in***.read();   } |

## 4.创建服务消费者项目

### 1）创建消费者项目

### 2）引入依赖

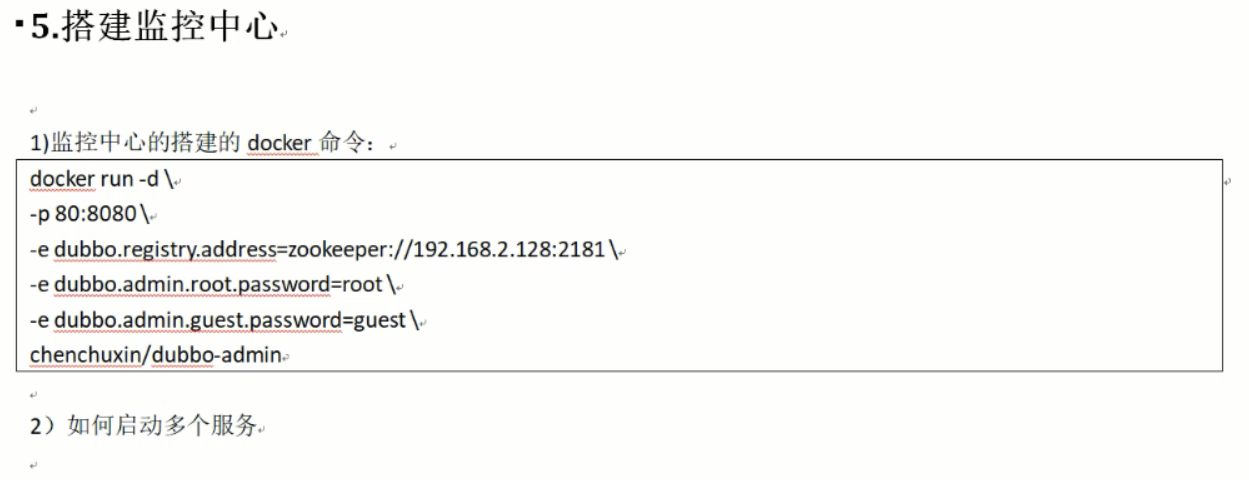
|  |
| --- |
| <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>dubbo</**artifactId**>  <**version**>2.5.8</**version**> </**dependency**> *<!-- zookeeper客户端依赖 -->* <**dependency**>  <**groupId**>com.github.sgroschupf</**groupId**>  <**artifactId**>zkclient</**artifactId**>  <**version**>0.1</**version**> </**dependency**> |

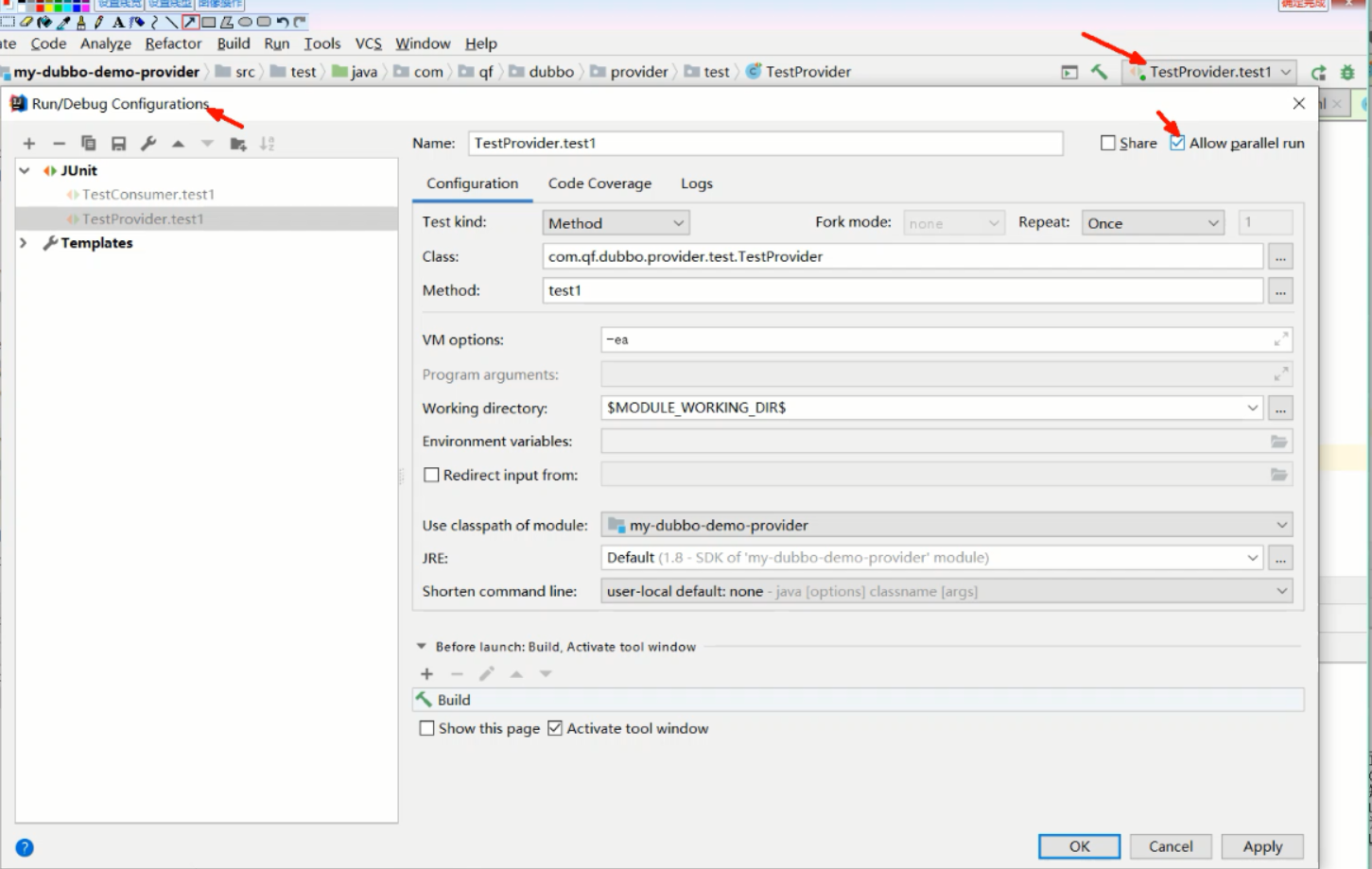
### 3）编写配置文件



### 4）编写测试方法，来调用服务提供者中的方法

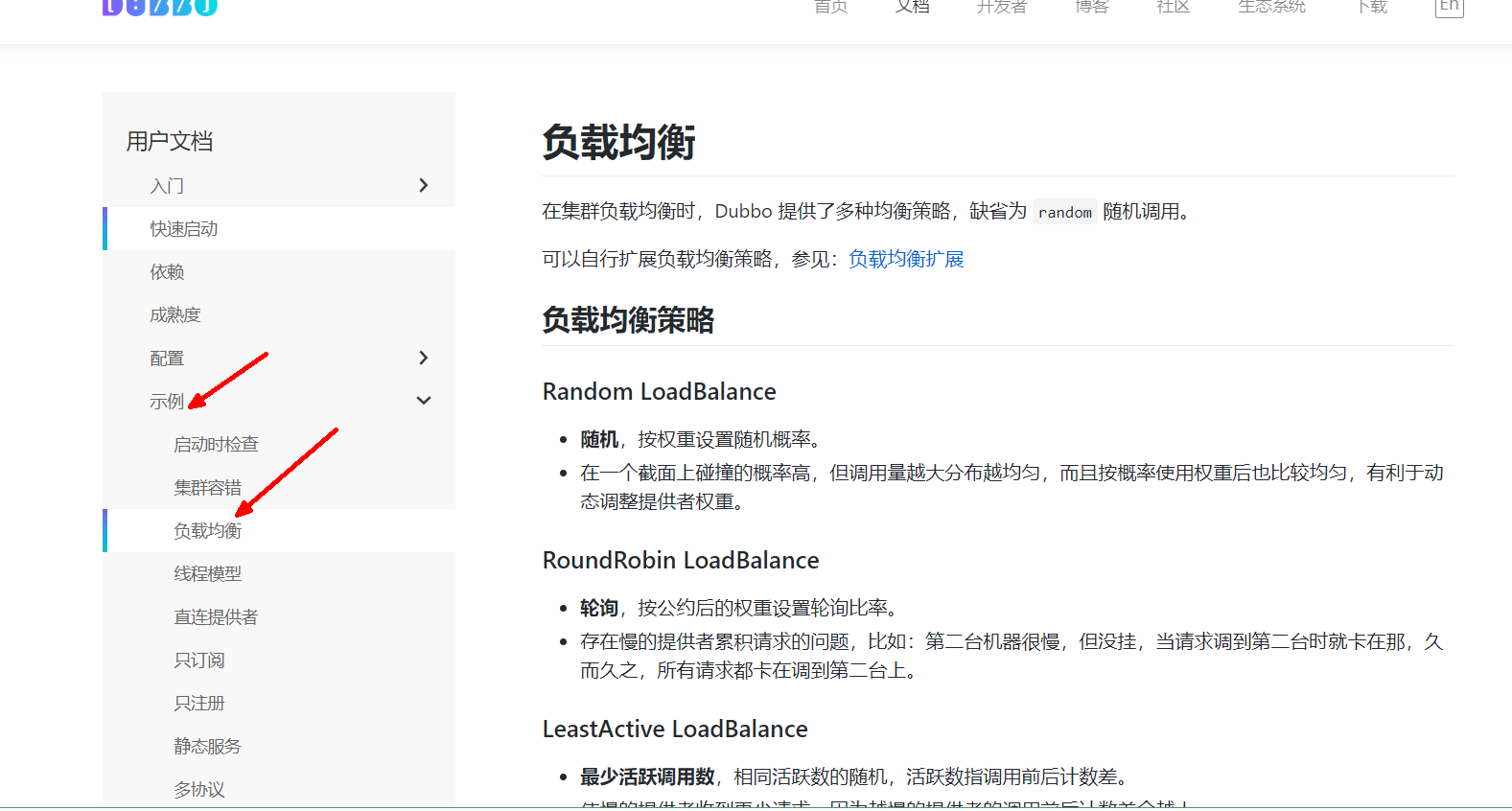
|  |
| --- |
| @Test **public void** test1(){   ClassPathXmlApplicationContext context =  **new** ClassPathXmlApplicationContext(**new** String[]{**"classpath:consumer.xml"**});   context.start();   *//获取接口的实现类的对象，实际上这个实现类的对象是dubbo提供的* IProductService productService = context.getBean(IProductService.**class**);   String product = productService.getProduct();  System.***out***.println(product);   } |

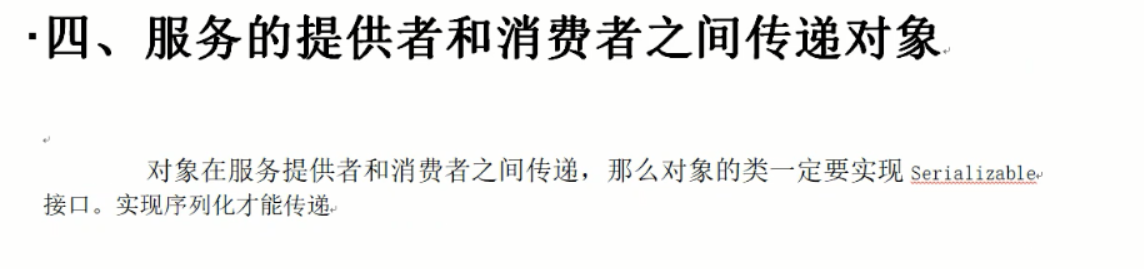




# 三、dubbo的负载均衡算法

参考官方文档：





# 四、Springboot整合dubbo-xml的方式

## 1.创建生产者（服务提供者）

1）创建项目后添加依赖

<**dependency**>  
 <**groupId**>com.alibaba.boot</**groupId**>  
 <**artifactId**>dubbo-spring-boot-starter</**artifactId**>  
 <**version**>0.2.0</**version**>  
</**dependency**>

2） 引入api项目的坐标

<**dependency**>  
 <**groupId**>com.qf</**groupId**>  
 <**artifactId**>my-dubbo-demo-api</**artifactId**>  
 <**version**>1.0.0-SNAPSHOT</**version**>  
</**dependency**>

3）编写接口的实现类

4）编写xml文件



5）在springboot启动程序上打上注解 @ImportResource()

@SpringBootApplication  
@ImportResource(**"classpath:provider.xml"**)  
**public class** MySpringBootDubboXmlProviderApplication {

监控中心的docker命令：

docker run -d \

-p 80:8080 \

-e dubbo.registry.address=zookeeper://192.168.2.128:2181 \

-e dubbo.admin.root.password=root \

-e dubbo.admin.guest.password=guest \

chenchuxin/dubbo-admin

## 2.创建服务消费者

1）引入依赖

2）编写配置文件

consumer.xml

3）在springboot上面加入注解，引入配置文件

@SpringBootApplication  
@ImportResource(**"classpath:consumer.xml"**)  
**public class** MySpringBootDubboXmlConsumerApplication {

4)测试消费者能否得到服务提供者提供的服务

|  |
| --- |
| @RunWith(SpringRunner.**class**) @SpringBootTest **public class** MySpringBootDubboXmlConsumerApplicationTests {    @Autowired  **private** IProductService **productService**;    @Test  **public void** contextLoads() {   String product = **productService**.getProduct();  System.***out***.println(product);    }  } |

# 五、Springboot整合dubbo-注解的方式

常用！

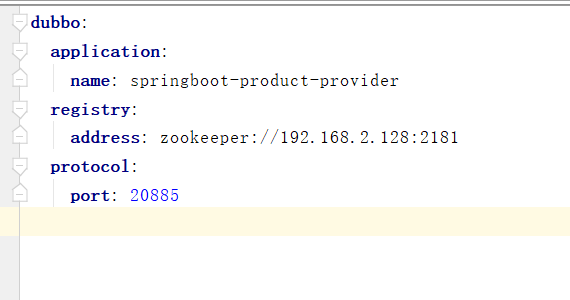
## 1.创建服务提供者

1）引入依赖： dubbo、api的

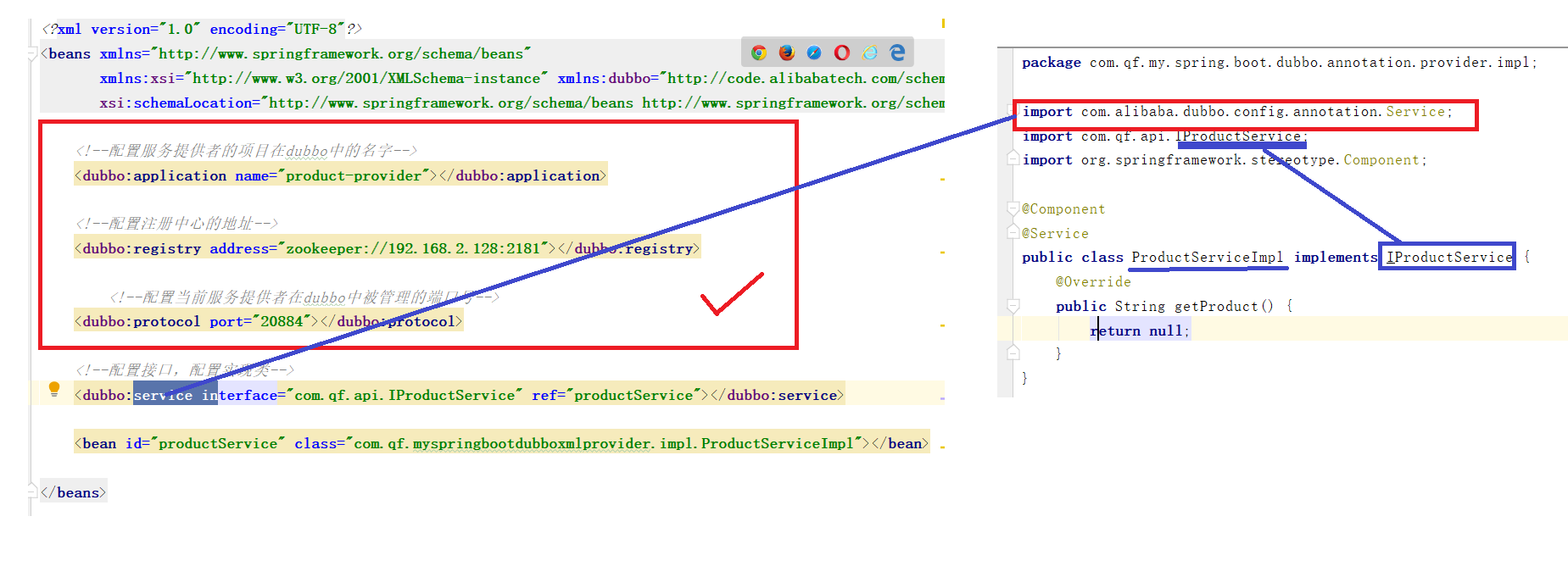
2）在springboot启动程序打上注解

@EnableDubbo

3）编写配置文件



4）在实现类上打上注解



## 2.服务消费者

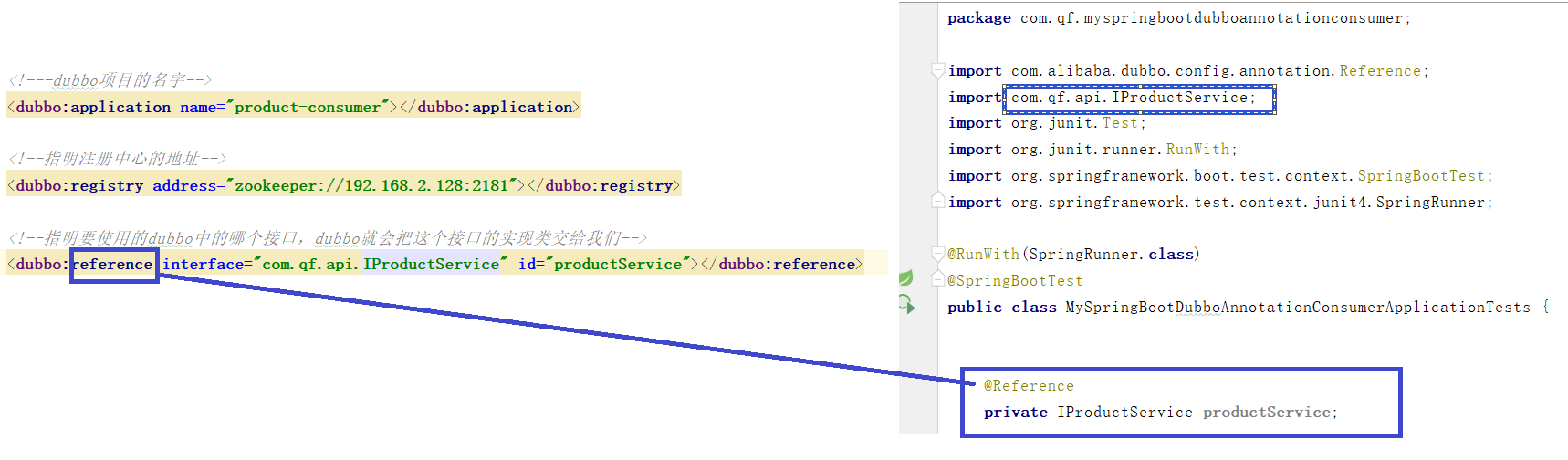
## 1）引入依赖

dubbo api

2）编写配置文件

3）springboot启动程序上启动dubbo

4）使用Reference注解来替代xml中的配置，达到从注册中心订阅服务的效果



# 六、逆向工程快速生成mybatis需要的东西